Fusing Prior Knowledge with Data-Driven Inferential Sensor Under Industrial Scenario

Zhichao Chen/Ziciu Can[1], **Hao Wang**[1], **Zhihuan Song**[1] **and Zhiqiang Ge**[2]
[1]Zhejiang University, [2]Pengcheng Laboratory

Accurate quality variable inference by process variables is the core of industrial inferential sensor modeling, where recent advancements have seen deep learning (DL) models achieving remarkable success. However, integrating knowledge of unit operations is critical for improving inferential sensor performance, yet it has received little attention. The main challenge lies in the incompleteness and correctness of industrial knowledge due to its semi-empirical nature and inevitable engineering errors. Addressing this, we introduce the Gradient Knowledge Network based on the graph neural network's message-passing mechanism within the variational Bayesian inference framework, which naturally copes with the abovementioned issues by fusing observational data. Initially, we parameterize the prior knowledge about the process variables, which mirrors the graph in graph neural network, as Dirichlet distribution based on the analysis of message passing mechanism. However, the divergence computation and normalization constraints are challenging for model implementation. To navigate these challenges, we transform the Bayesian inference problem into an optimization problem, subsequently recasting it as a simulation induced by the gradient field, ensuring compatibility with DL backends. Furthermore, we derive a theoretical iteration equation to maintain the normalization constraint. The architecture of our model and its learning algorithm are then detailed. Finally, we conduct various experiments on two real industrial processes to demonstrate the model's efficacy from the perspective of prediction accuracy, sensitivity analysis, and ablation study.

Keywords: Variational Inference

1. Introduction

Predicting hard-to-measure quality variable from high-dimensional easy-to-obtain sensory data plays a crucial role in intelligent manufacturing, supporting decision-making, and enhancing anomaly detection and diagnosis capabilities. Previous studies have explored various methodologies to realize this 'inferential sensor', including mechanistic equations and data-driven models [1–3], to tackle this challenge. However, as industrial processes grow more complex due to technological advancements, obtaining precise mechanistic equations has become increasingly difficult. Consequently, the focus has shifted towards data-driven models, appreciated for their simplicity and minimal reliance on prior knowledge.

The advancement of computational technologies and artificial intelligence has led to the application of deep learning (DL), especially neural networks, in analyzing industrial process data. Deep neural networks are classified into categories such as (stacked) auto-encoder [4], convolution neural network (CNN) [5], recurrent neural network [6], and Transformer [7, 8]. These models leverage feature extraction mechanisms, such as convolution operators in CNN exploiting local similarities and self-attention mechanism in Transformers utilizing permutation invariance, to autonomously extract features and facilitate the inference of quality variables based on these extracted features.

Nonetheless, there is room for improvement in the performance of DL-based models for inferential sensor modeling, particularly in terms of accuracy. Specifically, the data collected from the inferential

Corresponding author(s): zhiqiang.ge@hotmail.com

sensor must satisfy some (semi)-empirical equations since the equipment should be rigorously designed and rated based on these equations with engineering margins of 30% to 50%. This foundational knowledge shapes the data collected from process instrumentation, suggesting that its integration could markedly improve model performance. *On this basis, incorporating comprehensive, albeit incomplete, industrial process knowledge [9] from (semi)-empirical equations* into DL models offers a viable solution for enhancing prediction accuracy. A notable instance is the inferential sensor task in an absorber, designed to separate gas impurities based on solubility differences in the absorbent. In this context, accurately modeling the liquid-gas ratio, which is a critical factor determined during the absorber's design phase to predict the output gas concentration, and hence might significantly enhance the prediction accuracy of data-driven models. By embedding such specific process knowledge, particularly the relationship between the absorbent flow rate and gas flow rate that establishes the liquid-gas ratio, into DL models, we may substantially refine predictive accuracy.

However, merely possessing instrumental data is insufficient for developing DL-based inferential sensor models, particularly the thermodynamic data, crucial for computation, is not as readily available as in physical-informed neural networks [10]. In response, researchers have proposed utilizing graphs to represent the knowledge about relationships among process sensors. As shown in Fig. 1 (a), these models employ a graph, denoted as $\mathcal{A}$, to facilitate feature extraction by enabling covariates x to interact based on the graph structure, subsequently inferring the label y from these features. This approach contrasts with traditional data-driven inferential sensor startegy, where the goal is to predict y directly from x without considering $\mathcal{A}$.

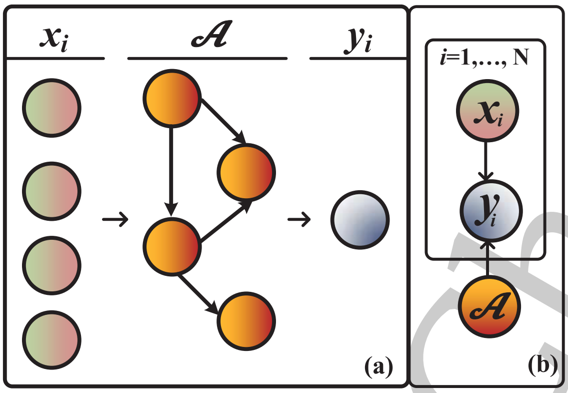

Figure 1 | The (a) illustration of GNNs and the (b) corresponding PGM.

Despite their successes, integrating prior knowledge into GNNs presents notable challenges. As illustrated in Fig. 1 (b), the learning objective of a GNN-based model, $p(y|x, \mathcal{A})$, differs fundamentally from that of an inferential sensor task, $p(y|x)$, indicating that $p(y|x) \neq p(y|x, \mathcal{A})$. Utilizing the Probabilistic Graphical Model (PGM) depicted in Fig. 1 (b) and applying Monte Carlo estimation, we approximate $p(y|x)$ as $\int p(y|x, \mathcal{A})p(\mathcal{A})\mathrm{d}\mathcal{A} \approx \frac{1}{\mathrm{n}}\sum_{i=1}^{n} p(y|x, \mathcal{A}_i)$, where n is sample number. Achieving accurate predictions is contingent upon effectively sampling from $p(\mathcal{A})$. Consequently, the direct employment of prior knowledge, i.e., directly adopting $\mathcal{A}$ via $\mathcal{A} \sim p(\mathcal{A})$, is fraught with issues listed as follows:

1. The incomplete property of the prior knowledge. The acquirement difficulty of prior knowledge from thermodynamic scale increases as the process complexity, which makes it difficult to obtain a complete and 'perfect' graph for predicting y based on the procedure given in Fig. 1 (a).
2. The correctness of the prior knowledge. For instance, fluids in industrial processes typically exhibit turbulence. Therefore, under varying conditions, assessed by certain immeasurable dimensionless Reynold's number, the appropriate modeling knowledge may differ, making prior knowledge

potentially incorrect in some scenarios.

Fortunately, the Bayesian framework offers a solution by suggesting the optimal $p(\mathcal{A})$ can be obtained by $p^*(\mathcal{A}) \propto p(\mathcal{A})p(y|\mathcal{A},x)$, i.e. a proper form is suggested to be $\mathcal{A} \sim p(\mathcal{A})p(y|\mathcal{A},x)$. Addressing the aforementioned challenges involves *'tolerating'* and *'correcting'* the prior $p(\mathcal{A})$ with the data-informed $p(y|x,\mathcal{A})$, ensuring that $\mathcal{A}$ used for testing is drawn from $p(\mathcal{A})p(y|\mathcal{A},x)$ rather than merely $p(\mathcal{A})$. This approach underscores the need for DL models in industrial process modeling to autonomously discover the most pertinent knowledge from the available prior information, moving beyond mere reliance on direct prior knowledge application.

In pursuit of this objective, an initial investigation into the representation of prior knowledge at the human cognition level within an industrial process context is conducted under the variational inference. This exploration reveals the limitations of traditional, parameter-based variational inference for DL backends [11, 12] implementation, chiefly due to the divergence computation and normalization constraints of Dirichlet distribution inherent in the GNNs. To overcome the divergence computation issue, we reformulate the challenge as an optimization problem, subsequently translating it into a partial differential equation (PDE) simulation problem induced by the 'gradient' field of target probability distribution. This leads to the derivation of a stochastic differential equation (SDE) form, which is more amenable to implementation using DL backend optimizers [13]. On this basis, to address the normalization constraint, this transformation facilitates the application of a mirror descent approach to ensure adherence to normalization constraints during SDE simulation. In summary, we propose our novel model named 'Gradient Knowledge Network' and its corresponding algorithm based on summarizing the abovementioned techniques. Finally, this paper concludes by demonstrating its efficacy through extensive experiments on two real-world inferential sensor tasks.

The novelty of this paper can be summarized as follows:

1. This work pioneers the embedding of uncertain prior knowledge pertaining to industrial processes into DL models for inferential sensor modeling and propose a novel model named GKN.
2. The paper addresses a critical hurdle: the intractability of divergence computation in variational inference problem within the DL backend due to the Dirichlet distribution. By theoretically reformulating the variational inference issue into a SDE simulation, it presents a solution compatible with DL backends.
3. Further innovating, the study introduces a mirror descent methodology tailored to ensure the normalization requirements of the Dirichlet distribution are met throughout the model's training process.

The rest of this manuscript is organzed as follows: To better understand paper, the preliminaries are introduced in Section 2. After that, we propose our core approach for fusing prior knowledge and data in Section 3. Consequently, we summarzie our model expressions and algorithm in Section 4, and conduct experimental analysis in Section 5. Finally, the conclusions are given in Section 6

2. Preliminaries

2.1. Message Passing and Graph Learning in GNN

According to Ma et al. [14], the message passing neural network (MPNN) is a general GNN framework. Graph convolution network [15], graph SAGE [16], and graph attention network [17] can be regarded as special cases of MPNN. The feature of node ν_i can be updated by (1) and (2) in MPNN.

$$m_i = \sum_{\nu_j \in \mathcal{D}(\nu_i)} \mathrm{Mess}(F_i, F_j, e_{(\nu_i,\nu_j)}) \tag{1}$$

$$F_i^{'} = \mathrm{Update}(F_i, m_i) \tag{2}$$

where Mess is the message function, e is the edge feature and Update stands for the update function. Through combining the original features and the aggregated message from its neighbors, the messages from the neighbors of node ν_i are generated by message function M and passed to node ν_i. By changing the summation operator, message function, and update function, the MPNN can be equivalent to other GNNs to extract spatial feature of process variables.

2.2. Variational Inference for Parameter & Wasserstein Space

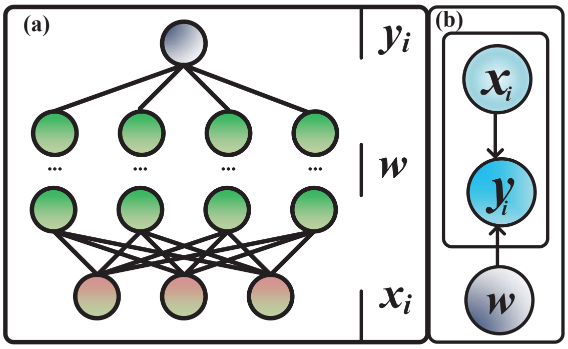

Figure 2 | The (a) illustration of neural network and the (b) corresponding PGM.

Adopting the notations from prior sections for clarity, we delve into Bayesian inference tasks as depicted in Fig. 2 (a) and (b). Classical variational inference methods strive to approximate the desired posterior $p(w|\mathcal{D})$ with a simpler distribution $q(w)$ that is easier to sample from. The objective is to minimize the KL divergence $\mathbb{D}_{\mathrm{KL}}[q(w)|p(w|\mathcal{D})]$ as defined by:

$$\underset{q(w)}{\arg\min} \quad \mathbb{D}_{\mathrm{KL}}\left[q(w)\|p(w|\mathcal{D})\right] \triangleq \int q(w)\log\frac{q(w)}{p(w|\mathcal{D})}\mathrm{d}w = \mathbb{E}_{q(w)}\left[\log\frac{q(w)}{p(w|\mathcal{D})}\right], \tag{3}$$

where $\mathbb{E}$ is the expectation operator.

However, the integral term is difficult to compute and thus, (3) can be further reformulated as follows:

$$\begin{aligned}\mathbb{D}_{\mathrm{KL}}(q(w)\|p(w|\mathcal{D})) &\overset{\text{(i)}}{=} \mathbb{E}_{q(w)}[\log\frac{q(w)p(\mathcal{D})}{p(\mathcal{D}|w)p(w)}] \\ &= -\underbrace{\mathbb{E}_{q(w)}\left[\log p(\mathcal{D}|w)\right]}_{\text{likelihood}} + \underbrace{\mathbb{D}_{\mathrm{KL}}\left[q(w)\|p(w)\right]}_{\text{regularization}} + \underbrace{\log p(\mathcal{D})}_{\text{constant}} \\ &\overset{\text{(ii)}}{=} -\mathbb{E}_{q(w)}\left[\log p(y|w,x)\right] + \mathbb{D}_{\mathrm{KL}}\left[q(w)\|p(w)\right],\end{aligned} \tag{4}$$

where (i) is based on the Bayesian equation, and (ii) is based on the fact that the dataset can be factorized according to the following equation for supervised task:

$$p(\mathcal{D}|w) = p(y|w,x)p(x). \tag{5}$$

Note that, $\log p(\mathcal{D})$ and $\mathbb{E}_{q(w)}\left[\log p(x)\right]$ are constants and thus dropped explicitly.

Drawing from (4) and the insights of Kingma et al. [18], the celebrated Bayesian inference of model parameter mainly consists of two parts, namely the update some belief over parameters w in the form of a prior distribution $p(w)$ under observational data $\mathcal{D}$, and update the belief over w in the form of posterior $p(\mathcal{D}|w)$. And thus, the constant term can be omitted explicitly and conventional loss function can be defined by the following equation:

$$\mathcal{L}_{\text{ELBO}} \triangleq -\mathbb{E}_{q(w)}\left[\log p(\mathcal{D}|w)\right] + \mathbb{D}_{\text{KL}}\left[q(w)\|p(w)\right]. \tag{6}$$

Note that, the probability distribution space of variational space is assumed to be Wasserstein space $\mathcal{P}_2(\mathbb{R}^d)$, where the probability distribution $p(\cdot) \in \mathcal{P}_2(\mathbb{R}^d)$ has finite second moment:

$$\mathcal{P}_2(\mathbb{R}^d) \triangleq \left\{p(\cdot)| \int_{\mathbb{R}^d} |x|^2 \mathrm{d}p(x) < \infty\right\}. \tag{7}$$

On this basis, let $\{p_t\}_{t\in[0,\tau]}$ be the set of absolutely continuous curves in $\mathcal{P}_2(\mathbb{R}^d)$. For $t \in [0,\tau]$, there exists a velocity field $v_t : \mathbb{R}^d \to \mathbb{R}^d$ which satisfies the celebrated continuity equation [19]:

$$\frac{\partial p_t(v_t)}{\partial t} + \nabla \cdot (v_t p_t(v_t)) = 0. \tag{8}$$

3. Proposed Approach

3.1. Description & Parameterization of Incomplete Knowledge

In Fig. 2 (b), the model parameters are *global* parameters, which are shared by all samples. It is interesting that the probability graph is similar to the probability graph of the GNNs in Fig, 1 (b). This phenomenon indicates that prior knowledge may also be encoded as the *belief* into the data-driven model. And thus, the task in this subsection is to parameterize the prior knowledge for further encoding operation.

Before knowledge parameteriztion, the first problem is how to describe the graph with proper distribution to obtain the graph that undertakes the knowledge. The next issue is how to bridge the human cognition to the industrial process and the distribution parameters. Thereby, the rest of this subsection will focus on how to describe the prior graph and parameterize it.

3.1.1. Description of Graph

- **Mathematical Description:** How to select a proper distribution for graph description?
- **Cognition Connection:** How to connect the distribution parameters with human cognition degree to industrial process?

Intuitively, according to Ying et. al [20], the mean-field assumption on the graph is the natural choice for graph decomposition out of simplicity. Based on this result, the prior can be factorized as (9) follows:

$$p(\mathcal{A}) = \prod_{i=1}^{N}\prod_{j=1}^{N} p(\alpha_{ij}^{o}), \tag{9}$$

where the α illustrates the edge of the graph, and the edges of the prior knowledge graph may have two states namely presence or absence, respectively. As such, the α can be regarded as *binary variable* and denoted as $\alpha \in \{0,1\}$. On this basis, Bernoulli distribution can be adopted to describe the existence probability of edge as (10):

$$p(\alpha_{i,j}^{o}) = p_{\text{Bern}}(\alpha_{i,j}^{o}|\rho_{ij}^{o}), \tag{10}$$

where the $\rho \in [0, 1]$ stands for the parameter of Bernoulli distribution.

Throughout the above-mentioned operation, ‘mathematical description’ can be addressed. After that, it remains a challenge for how to bridge the gap between the human cognition degreee and the parameter of Bernoulli distribution. According to the cognition degree of the industrial process, the logical parameters $\mathbf{Si}, \mathbf{i} \in \{1, 2, 3\}$ can be defined to portray the relationship between industrial process knowledge and model parameters:
S1: there exist prior knowledge between the process variables i and j
S2: the existence of prior knowledge about the process variables i and j is unknown.
S3: there exist no prior knowledge between the process variables i and j.
Hence, the expressions can be given in (11):

$$\begin{bmatrix} \mathbf{S1} \\ \rho^o = 0.955 \end{bmatrix} \veebar \begin{bmatrix} \mathbf{S2} \\ \rho^o = 0.500 \end{bmatrix} \veebar \begin{bmatrix} \mathbf{S3} \\ \rho^o = 0.045 \end{bmatrix} \tag{11}$$

where the value ρ^o is determined by the $2 - \sigma$ rule, and $\veebar$ is the disjunctive operation indicates that only one condition may take effect. In summary, the abovementioned operation address the “cognition connection” problem.

3.1.2. Parameterization of Graph Under Message Passing Framework

Merely obtaining the graph $\mathcal{A}$ is insufficient for feature extraction due to the requisite normalization step that follows graph acquisition. Consequently, the graph is parameterized on a row-by-row basis as follows:

$$\hat{\mathcal{A}} = D^{-\frac{1}{2}} \mathcal{A} D^{-\frac{1}{2}}, \tag{12}$$

where the diagonal matrix D represents the degree matrix of graph $\mathcal{A}$, defined by:

$$D \triangleq \mathrm{diag}(\sum_{i=1}^{\mathrm{N_{node}}} \mathcal{A}_{i,j}), \tag{13}$$

with $\mathrm{N_{node}}$ denoting the number of edges.

For instance, given $\mathcal{A}$ as $\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$, D would be calculated as $\begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix}$, resulting in $\hat{\mathcal{A}}$ being $\begin{bmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 1 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}$. This analysis elucidates that the required graph for feature extraction is $\hat{\mathcal{A}}$, as opposed to the unnormalized $\mathcal{A}$. Significantly, the sum of each row in $\hat{\mathcal{A}}$ is equal to 1. Consequently, the prior distribution of $\hat{\mathcal{A}}$ is revised to:

$$p(\hat{\mathcal{A}}) = \prod_{i=1}^{\mathrm{N_{node}}} p(\vec{\alpha}_i), \tag{14}$$

where $\vec{\alpha}_i \in \mathbb{R}^{1 \times \mathrm{N_{node}}}$ is specified as:

$$\vec{\alpha}_i = \left[\alpha_{i,1}, \ldots, \alpha_{i,\mathrm{N_{node}}}\right], \sum_{j=1}^{\mathrm{N_{node}}} \alpha_{i,j} = 1. \tag{15}$$

From (14) and (15), it is apparent that each row of $\hat{\mathcal{A}}$ adheres to the Dirichlet distribution, and thus $\hat{\mathcal{A}}$ can be represented as:

$$q(\hat{\mathcal{A}}) = \prod_{i=1}^{\mathrm{N_{node}}} p(\vec{\alpha}_i), \vec{\alpha}_i \sim \mathrm{Dir}(\vec{\rho}_i) = \frac{\Gamma(\sum_{j=1}^{\mathrm{N_{node}}} \rho_{i,j})}{\prod_{j=1}^{\mathrm{N_{node}}} \Gamma(\rho_{i,j})} \prod_{j=1}^{\mathrm{N_{node}}} \alpha_{i,j}^{\rho_{i,j}-1}, \tag{16}$$

where $\Gamma(\cdot)$ is the gamma function, defined as:

$$\Gamma(\rho) \triangleq \rho e^{\gamma\rho} \prod_{n=1}^{\infty} \left[\left(1+\frac{\rho}{n}\right) \exp\left(-\frac{\rho}{n}\right) \right], \tag{17}$$

γ represents Euler's constant, and $\vec{\rho}$ is the concentration parameter, set as per (18):

$$\vec{\rho}_i = \left[\rho^o_{i,1}, \ldots, \rho^o_{i,\mathrm{N}_{\mathrm{node}}} \right]. \tag{18}$$

3.2. Inference of $p(\hat{A}|\mathcal{D})$

3.2.1. Infeasibility of Directly Parameterization $q(\hat{A})$

In this subsection, our primary focus is on the inference of $q(\hat{\mathcal{A}})$. Direct application of (16) and (18) into (6) for model training introduces computational challenges due to the intractability of the KL divergence between Dirichlet distributions. Specifically, for distributions $P \sim \mathrm{Dir}(\rho_1)$ and $Q \sim \mathrm{Dir}(\rho_2)$, the KL divergence is expressed as:

$$\mathbb{D}_{\mathrm{KL}}[P\|Q] = \ln \frac{\Gamma\left(\sum_{i=1}^{k} \rho_{1i}\right)}{\Gamma\left(\sum_{i=1}^{k} \rho_{2i}\right)} + \sum_{i=1}^{k} \ln \frac{\Gamma(\rho_{2i})}{\Gamma(\rho_{1i})} + \sum_{i=1}^{k} (\rho_{1i} - \rho_{2i}) \left[\Psi(\rho_{1i}) - \Psi\left(\sum_{i=1}^{k} \rho_{1i} \right) \right], \tag{19}$$

where $\Psi(\cdot)$, the psi function, is defined as:

$$\Psi(\rho) \triangleq \frac{\Gamma(\rho)}{\nabla_\rho \Gamma(\rho)}. \tag{20}$$

Given this analysis, directly deriving $q(\hat{\mathcal{A}})$ from (6) proves to be infeasible due to the complexity of computing the KL divergence between two Dirichlet distributions. It is essential to emphasize that our objective is to sample $\hat{\mathcal{A}}$ for model training & testing, not to obtain the distribution $q^*(\hat{\mathcal{A}})$ directly. Therefore, in the subsequent part of this subsection, we propose representing $q^*(\hat{\mathcal{A}})$ *by* $\hat{\mathcal{A}}$ *and deriving a practical approach to sample* $\hat{\mathcal{A}}$ *from posterior distribution* $p(\hat{\mathcal{A}}|\mathcal{D})$ *within DL backends.*

3.2.2. Variational Inference as Simulation Problem

To reframe the problem identified in (3), let us consider it as a functional optimization problem, leveraging the definition provided by (7):

$$\underset{q(\hat{\mathcal{A}})\in\mathcal{P}2(\mathbb{R}^d)}{\arg\min} \quad \mathcal{F} \triangleq \mathbb{D}_{\mathrm{KL}}[q(\hat{\mathcal{A}})\|p(\hat{\mathcal{A}}|\mathcal{D})]. \tag{21}$$

In alignment with (8), by conceptualizing the optimization of $q(\hat{\mathcal{A}})$ over the interval $t \in [0, \tau]$ as a dynamic curve $q_t(\hat{\mathcal{A}})$ within the Wasserstein space, the optimal solution $q^*(\hat{\mathcal{A}})$ is attainable at the terminal point τ. Building upon this conceptual framework, we introduce the following lemma:

Lemma 1. *The curve* $q_t(\hat{\mathcal{A}})$ *can reduce* $\mathcal{F}$ *as long as the following equation holds:*

$$v_t = -\nabla \frac{\delta \mathcal{F}[q_t(\hat{\mathcal{A}})]}{\delta q_t(\hat{\mathcal{A}})}, \tag{22}$$

where δ *is first variation.*

On this basis, the following proposition for optimizing (21) can be given as follows:

Proposition 3.1. *The velocity v_t for optimizing (21) is defined as follows:*

$$v_t = -\nabla_{\hat{\mathcal{A}}} \frac{\delta \mathbb{D}_{\mathrm{KL}} \left[q_t(\hat{\mathcal{A}}) \| p(\hat{\mathcal{A}}|\mathcal{D}) \right]}{\delta q_t(\hat{\mathcal{A}})} = \nabla_{\hat{\mathcal{A}}} \log p(\mathcal{A}\hat{|}\mathcal{D}) - \nabla_{\hat{\mathcal{A}}} \log q_t(\hat{\mathcal{A}}). \tag{23}$$

Following the proposition, incorporating (23) into (8) renders the optimization of $q(\hat{\mathcal{A}})$ analogous to simulating the PDE that is induced by the gradient field $\nabla_{\hat{\mathcal{A}}} \log p(\hat{\mathcal{A}}|\mathcal{D})$:

$$\frac{\partial q_t(\hat{\mathcal{A}})}{\partial t} + \nabla_{\hat{\mathcal{A}}} \cdot [(\nabla_{\hat{\mathcal{A}}} \log p(\hat{\mathcal{A}}|\mathcal{D}) - \nabla_{\hat{\mathcal{A}}} \log q_t(\hat{\mathcal{A}}))q_t(\hat{\mathcal{A}})] = 0. \tag{24}$$

3.2.3. Implementation Feasibility Under DL backends

Building upon (6), we have derived a simplified approach for constructing a graph aimed at minimizing the KL divergence, denoted as $\mathbb{D}_{\mathrm{KL}}[q(\hat{\mathcal{A}}) \| p(\hat{\mathcal{A}}|\mathcal{D})]$. In this subsection, our objective is to elucidate the practicality of its implementation within DL backends. To this end, we present the following lemma:

Lemma 2 (Theorem 5.4 in Reference [21])**.** *For a stochastic differential equation given as follows:*

$$\mathrm{d}x_t = f_t(x_t)\mathrm{d}t + g_t \mathrm{d}W_t, \tag{25}$$

where $f(x_t)$, g_t, and $\mathrm{d}W_t$ are drifting term, volialty term, and Wiener process respectively. Its probability density $p_t(x_t)$ can be given by the following equation known as Fokker-Planck equation:

$$\frac{\partial p_t(x_t)}{\partial t} = -\nabla_x(p_t(x_t)f_t(x_t)) + \frac{1}{2} g_t^2 \nabla_x \cdot \nabla_x p_t(x_t). \tag{26}$$

Drawing upon Lemma 2, by integrating (23) into (8), the density equation for $q_t(\hat{\mathcal{A}})$ is formulated as:

$$\frac{\partial q_t(\hat{\mathcal{A}})}{\partial t} = \nabla_{\hat{\mathcal{A}}} \cdot [q_t(\hat{\mathcal{A}}) \nabla_{\hat{\mathcal{A}}} \log p(\hat{\mathcal{A}}|\mathcal{D})] - \nabla_{\hat{\mathcal{A}}} \cdot \nabla_{\hat{\mathcal{A}}} q_t(\hat{\mathcal{A}}). \tag{27}$$

Comparing (27) with (26) yields:

$$\mathrm{d}\hat{\mathcal{A}} = -\nabla_{\hat{\mathcal{A}}} \log p(\hat{\mathcal{A}}|\mathcal{D})\mathrm{d}t + \sqrt{2}\mathrm{d}W_t. \tag{28}$$

Applying the Euler-Maruyama method to (28) with a step size η [22, 23] results in:

$$\hat{\mathcal{A}}^{t+1} = \hat{\mathcal{A}}^t - \eta \nabla_{\hat{\mathcal{A}}} \log p(\hat{\mathcal{A}}|\mathcal{D})|_{\hat{\mathcal{A}}=\hat{\mathcal{A}}t} + \sqrt{2\eta} \mathcal{N}(0, I), \tag{29}$$

where the gradient $\nabla \hat{\mathcal{A}} \log p(\hat{\mathcal{A}}|\mathcal{D})$ is decomposed as:

$$\nabla_{\hat{\mathcal{A}}} \log p(\hat{\mathcal{A}}|\mathcal{D}). = \nabla_{\hat{\mathcal{A}}} [\log p(\mathcal{D}|\hat{\mathcal{A}}) + \log p(\hat{\mathcal{A}}) - \log p(\mathcal{D})] \overset{\text{(i)}}{=} \nabla_{\hat{\mathcal{A}}} [\underbrace{\log p(y|\hat{\mathcal{A}}, x)}_{\text{log-likelihood}} + \underbrace{\log p(\hat{\mathcal{A}})}_{\text{prior distribution}}], \tag{30}$$

with (i) leveraging the fact that $\nabla_{\hat{\mathcal{A}}} \log p(\mathcal{D}) = 0$ and $\nabla_{\hat{\mathcal{A}}} \log p(x) = 0$. Furthermore, (29) aligns with the conventional iteration form of optimizers [13] utilized in DL backends, where step size η is referred to as the learning rate, and the 'log-likelihood' term can be implemented by mean square error loss. This demonstrates the practicality of learning $\hat{\mathcal{A}}$ using optimizers available in DL backends.

3.3. Simplex Constraint

In the preceding subsection, we completed the posterior inference of $\hat{\mathcal{A}}$ by reformulating it as an optimization problem solvable using optimizers available in DL backends. However, it is imperative during the optimization process to ensure that the row sum of $\hat{\mathcal{A}}$ equals 1. To address this requirement, we further refine the problem formulation, drawing upon (28) and (16), as detailed below:

$$\begin{aligned} \min_{\hat{\mathcal{A}}} \quad & -\log p(\hat{\mathcal{A}}|\mathcal{D}) \\ s.t. \quad & \begin{cases} \hat{\mathcal{A}} = \prod_{i=1}^{\mathrm{N_{node}}} p(\vec{\alpha}_i), \quad \vec{\alpha}_i \sim \mathrm{Dir}(\vec{\rho}_i) \\ \vec{\alpha}_i = \left[\alpha_{i,1}, \ldots, \alpha_{i,\mathrm{N_{node}}}\right] \\ \sum_{j=1}^{\mathrm{N_{node}}} \alpha_{i,j} = 1, \alpha_{i,j} \in \mathbb{R}^+ \end{cases} \end{aligned} \tag{31}$$

To promise the constraint on a simplex, we have the following proposition:

Proposition 3.2. *The row vector $\vec{\alpha}_i$ can be optimized via the following equations:*

$$\begin{cases} \log \vec{\alpha}_i^{t+1} = \log \vec{\alpha}_i^t - \eta \nabla_{\vec{\alpha}_i} \log p(\hat{\mathcal{A}}^t|\mathcal{D}) + \lambda_i \\ \lambda_i = -\log\{\sum_{j=1}^{N_{node}} \alpha_{i,j} \exp[-\eta \cdot (\nabla_{\vec{\alpha}_i} \log(\hat{\mathcal{A}}^t|\mathcal{D}))_{i,j}]\} \end{cases} \tag{32}$$

3.4. Implementation Under DL Backends

Eq. (32) introduces the use of the logarithm operator on $\log \vec{\alpha}_i$, alongside the application of the gradient operator on the loss function with respect to $\vec{\alpha}_i$ (denoted as $\nabla \vec{\alpha}_i$). For the sake of simplicity, we designate $\log \vec{\alpha}_i$ as a model parameter, and consequently, the iteration formula for $\log \vec{\alpha}_i$, leveraging the chain rule $\nabla_x f(x) = \frac{1}{x} \nabla_{\log(x)} f[\log(x)]$, is articulated as follows:

$$\log \vec{\alpha}_i^{t+1} = \log \vec{\alpha}_i^t - \frac{\eta}{\vec{\alpha}_i} \nabla_{\log \vec{\alpha}_i} \log p(\hat{\mathcal{A}}^t|\mathcal{D}). \tag{33}$$

Subsequently, the normalization step described in (32) is effectively executed through the application of the log-softmax operator (denoted as LogSoftmax):

$$\log \vec{\alpha}_i^{t+1} = \mathrm{LogSoftmax}[\log \vec{\alpha}_i^t - \frac{\eta}{\vec{\alpha}_i} \nabla_{\log \vec{\alpha}_i} \log p(\hat{\mathcal{A}}^t|\mathcal{D})]. \tag{34}$$

4. Model Implementation & Discussions

Drawing from the discussions in Sections 2.1 and 3, we present the visual depictions of our model in Fig. 3. Given its significant reliance on the simulation of gradient field, our model is aptly termed the ‘Gradient Knowledge Network’ (GKN). The subsequent parts of this section are structured in alignment with the framework outlined in Fig. 3.

4.1. Model Architecture Illustration

4.1.1. Model Overview

From Fig. 3 (a), our model can be divided into two parts namely ‘message passing’ and ‘label prediction’ in the rest of this subsubsection, we will introduce these two parts in detailed.

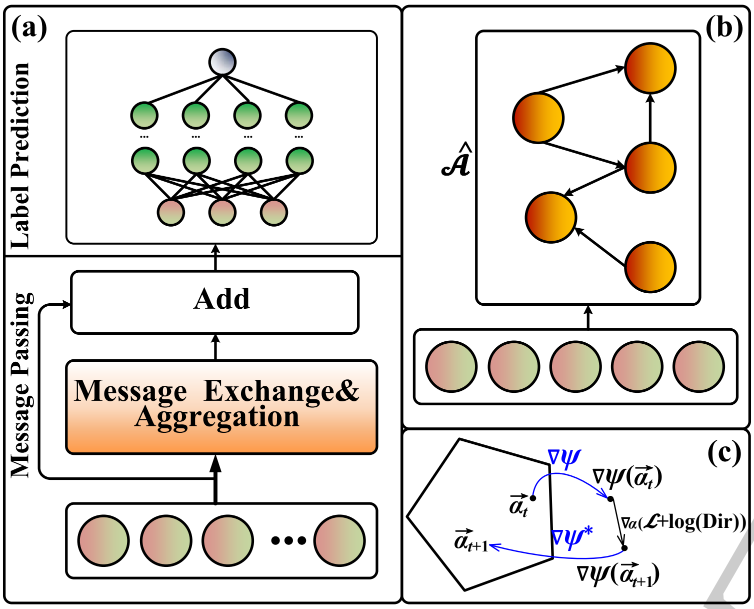

Figure 3 | The illustrations of (a) Gradient Knowledge Network, (b) forward computation process of Message Exchange & Aggregation Block, (c) optimization procedure of $\hat{\mathcal{A}}$, the polygon indicates that the solution space of $\hat{\mathcal{A}}$ is on a simplex.

- **Message Passing:** The model's architecture is elaborated upon in Section 2.1, where the input covariate vector is denoted as $x \in \mathbb{R}^{1\times \mathrm{N_{node}}}$. In the message passing component, each dimension of the covariates undergoes an initial transformation via an embedding layer, expressed as:

$$g_i^{\mathrm{embed}} = x_i \times W_i^{\mathrm{embed}} + b_i^{\mathrm{embed}}|_{i=1}^{\mathrm{N_{node}}}, \tag{35}$$

where embedding feature $g^{\mathrm{embed}} \in \mathbb{R}^{1\times \mathrm{N_{node}}}$. After that, the message is passed through the normalized graph matrix $\hat{A}$ to conduct the 'Message Exchange & Aggregation Operation' as we indicate in Fig. 3 (b):

$$g^{\mathrm{mess}} = g^{\mathrm{embed}}\hat{\mathcal{A}}W^{\mathrm{mess}} + b^{\mathrm{mess}}. \tag{36}$$

Notabaly, each row of $\hat{\mathcal{A}}$ is denoted by $\vec{\alpha}_i$ which is computed by the following equation:

$$\vec{\alpha}_i = \exp(\log\vec{\alpha}_i). \tag{37}$$

On this basis, the feature is updated as follows:

$$g^{\mathrm{update}} = g^{\mathrm{mess}} + xW^{\mathrm{id}} + b^{\mathrm{id}}, \tag{38}$$

where superscript id indicates identity.

- **Label Prediction:** Utilizing the features derived from the message passing component, the label prediction segment is implemented through a multi-layer perceptron (MLP) with a length of L. Consequently, the label y is predicted in the following manner:

$$\hat{y} = \left\{\left[\ldots\sigma(g^{\mathrm{update}}W^1 + b^1)\right]W^{\mathrm{L}} + b^{\mathrm{L}}\right\}, \tag{39}$$

where the superscript indicates the layer number, and σ is the activated function. In this study we set is as the ReLU function:

$$\sigma(x) \triangleq \max\{0, x\}. \tag{40}$$

4.1.2. Complexity Analysis

Based on previous subsection, in this part, we propose the complexity analysis as follows:

- **Time Complexity**: Denote D as feature dimension, and its superscript as layer number. Based on previous subsubsection $\mathrm{D}^0 = \mathrm{N}_{\text{node}}$. On this basis, the time complexity of our model can be summarized as follows:

$$\underbrace{\mathcal{O}\{\mathrm{N}_{\text{node}} + 2\mathrm{N}_{\text{node}}^2\}}_{\text{Message Passing}} + \underbrace{\sum_{l=1}^{\mathrm{L}-1} \mathcal{O}\left\{\mathrm{D}^l \mathrm{D}^{l+1}\right\}}_{\text{Label Prediction}}. \tag{41}$$

- **Memory Complexity**: The memory complexity of our model can be summarized as follows:

$$\underbrace{\mathcal{O}\{3\mathrm{N}_{\text{node}} + 2\mathrm{N}_{\text{node}}^2\}}_{\text{Message Passing}} + \underbrace{\sum_{l=1}^{\mathrm{L}-1} \mathcal{O}\left\{\mathrm{D}^l \mathrm{D}^{l+1}\right\} + \mathcal{O}\left\{\mathrm{D}^{l+1}\right\}}_{\text{Label Prediction}}. \tag{42}$$

4.2. Ovearll Algorithm

Algorithm 1 outlines the forward computation process of the GKN model, while Algorithm 2 details its training methodology. A distinctive feature of the GKN, as revealed from lines 7 to 10 in Algorithm 2, the update of $\hat{\mathcal{A}}$ adheres to the formulations presented in Section 3.4. These formulations ensure the simplex constraint of the row vector within $\hat{\mathcal{A}}$, marking a departure from prior methodologies. The illustration of this procedure are visually depicted in Fig. 3 (c).

Algorithm 1: The GKN model inference Algorithm

```
Input: Covaraite: ${x_1, x_2, ..., x_n}$
Parameter: Model Parameter set $\theta$
Output: Prediction value:${\hat{y}_1, \hat{y}_2, ..., \hat{y}_n}$
Message Passing Part:
/* $\hat{\mathcal{A}}$ formulation                                  */
$\vec{\alpha}_i \leftarrow \exp(\log \vec{\alpha}_i)$;
/* Message Exchange                                   */
$g_i^{\text{embed}} \leftarrow x_i \times W_i^{\text{embed}} + b_i^{\text{embed}}|_{i=1}^{\mathrm{N}_{\text{node}}}$;
$g^{\text{mess}} \leftarrow g^{\text{embed}} \hat{\mathcal{A}} W^{\text{mess}} + b^{\text{mess}}$;
/* Message Aggregation                                */
$g^{\text{update}} \leftarrow g^{\text{mess}} + xW^{\text{id}} + b^{\text{id}}$;
Label Prediction Part:
$\hat{y} \leftarrow \{[...\sigma(g^{\text{update}}W^1 + b^1)]\, W^{\mathrm{L}} + b^{\mathrm{L}}\}$
```

Algorithm 2: Training & Testing Algorithm

```
Input: Train, validate, and test data: $\{x_m, y_m\}_{m\in\mathcal{D}_{\text{train}}}$, $\{x_v, y_v\}_{v\in\mathcal{D}_{\text{valid}}}$, $\{x_n, y_n\}_{n\in\mathcal{D}_{\text{test}}}$
Hyperparameters:
Batch size: $\mathcal{B}$, learning Rate: $\eta$, Epoch: $\mathcal{E}$, parameter $\theta$ (note that $\log\vec{\alpha}\in\theta$), Prior Graph
(parameter of Dirichlet distribution) $\mathcal{A}^o$: $[\vec{\rho}_1^\top, ..., \vec{\rho}_{\text{N}_{\text{node}}}^\top]^\top$
Training:
Set epoch = 1
while epoch $\le \mathcal{E}$ do
    /* Forward Computation */
    Sample a minibatch $\mathcal{D}_{\text{minibatch}} \subset \mathcal{D}_{\text{train}}$;
    $\{\hat{y}_i\}_{i=1}^{\mathcal{B}} \leftarrow$ Algorithm 1;
    /* Computation of $\log p(y|\hat{\mathcal{A}}, x)$ */
    $\mathcal{L} \leftarrow \frac{1}{\mathcal{B}} \sum_{i=1}^{\mathcal{B}} (y_i - \hat{y}_i)^2$
    /* Gradient with respect to $\theta$ */
    Obtain gradient: $\nabla_\theta \mathcal{L}$
    /* Modify Gradient $\log\vec{\alpha}_i$ */
    $\nabla_{\log\vec{\alpha}_i} \log p(\hat{\mathcal{A}}|\mathcal{D}) \leftarrow \nabla_{\log\vec{\alpha}_i} [\log p(y|x, \hat{\mathcal{A}}) + \log p_{\mathcal{A}^o}(\hat{\mathcal{A}})]$
    $\nabla_{\log\vec{\alpha}_i} \log p(\hat{\mathcal{A}}|\mathcal{D}) \leftarrow \frac{1}{\vec{\alpha}_i} \nabla_{\log\vec{\alpha}_i} \log p(\hat{\mathcal{A}}|\mathcal{D})$
    Update model parameter $\theta$ with gradient under learning rate $\eta$
    $\log\vec{\alpha}_i \leftarrow$ LogSoftmax $\log\vec{\alpha}_i$
end
Save the best estimated model $\hat{\theta} = \theta_{\text{best}}$ on the validate dataset with min $\sum_{b=1}^{\text{N}_{\text{test}}} (\hat{y}_b - y_b)^2$.
Testing:
Predict the testing data set with Algorithm 1 with parameter $\hat{\theta}$.
```

5. Experimental Results

5.1. Comparison of Different Baseline Models

5.1.1. Datasets

We select the Carbon-Dioxide Absorber (CA) and Primary Reformer (PR) as dataset to demonstrate the effectiveness of GKN model. The flowsheets are given in Fig. 4 and 5, respectively. Based on the abovementioned introduction, the prior knowledge about these two processes is given as Fig. 6 based on (11). More detailed descriptions about these two processes and the derivation of prior knowledge are given in Section S.I supplementary material.

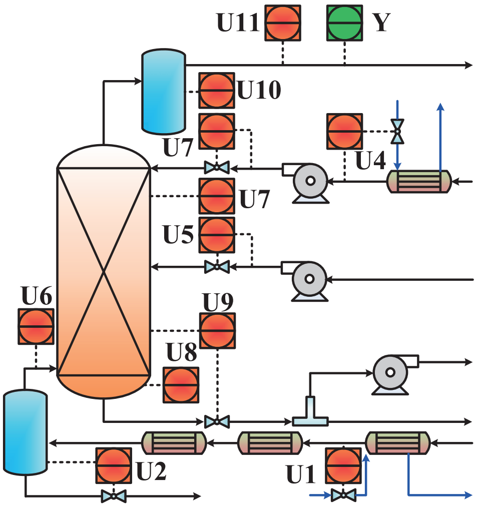

Figure 4 | Flowsheet of CA.

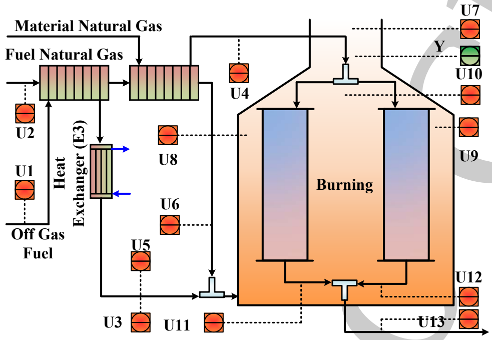

Figure 5 | Flowsheet of PR.

5.1.2. Baseline Selection

To provide a thorough comparison and showcase the effectiveness of our model, we have chosen to benchmark it against an array of baseline models categorized as follows:

- **Input Similarity-based Graph:** Flowformer (local) [24] and iTransformer (local) [25]
- **Learning-based Graph:** Random Synthesizer (Synthesizer(Random), local) [26] and stacked graph convolutional network (S-GCN, global) [27]

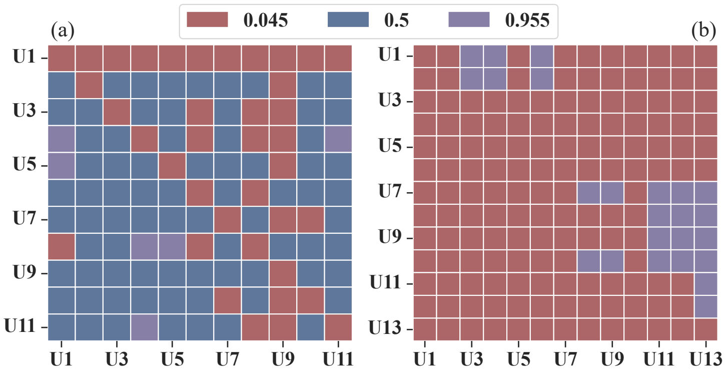

Figure 6 | Prior knowledge of (a) CA, (b) PR.

- **Fixed Graph:** Graph Convolution Network (GCN, global) [28], Graph Attention (GAT, global) [29], Graph Attention V2 (GAT-V2, global) [30]

Notably, *the iTransformer is the state-of-the-art model published in 'The Twelfth International Conference on Learning Representations' (ICLR-2024) in time-series forecasting task*. The reasons for choosing these models and hyperparameters are provided in Section S.II supplementary material.

5.1.3. *Evaluation Metrics*

To evaluate the performance of the inferential sensor model, the RMSE, R^2, MAE, and MAPE in (43) to (46) are adopted, where the $\mathrm{N}_{\mathrm{test}}$ and $\bar{y}$ are the size of the tesing dataset and average value, respectively. For RMSE, MAE, and MAPE, the closer value is to 0, the more accurate the prediction. While for R^2, the closer R^2 is to 1, the better the prediction performance of model.

$$\mathrm{RMSE} = \sqrt{\frac{1}{\mathrm{N}_{\mathrm{test}}} \sum_{l=1}^{\mathrm{N}_{\mathrm{test}}} (y_l - \hat{y}_l)^2} \tag{43}$$

$$\mathrm{R}^2 = 1 - \frac{\sum_{l=1}^{\mathrm{N}_{\mathrm{test}}} (y_l - \hat{y}_l)^2}{\sum_{l=1}^{\mathrm{N}_{\mathrm{test}}} (y_l - \bar{y})^2} \tag{44}$$

$$\mathrm{MAPE} = \frac{1}{\mathrm{N}_{\mathrm{test}}} \sum_{l=1}^{\mathrm{N}_{\mathrm{test}}} |\frac{(y_l - \hat{y}_l)}{y_l}| \times 100\% \tag{45}$$

$$\mathrm{MAE} = \frac{1}{\mathrm{N}_{\mathrm{test}}} \sum_{l=1}^{\mathrm{N}_{\mathrm{test}}} |y_l - \hat{y}_l| \tag{46}$$

5.1.4. *Prediction Performance Comparison*

Table 1 showcases the comparative performance of our model against various baseline models, yielding several insightful observations:

Table 1 | Overall Performance of GKN and Competitive Baselines

Model	CA				PR			
	R^2	RMSE	MAPE	MAE	R^2	RMSE	MAPE	MAE
Flowformer	0.74686	0.00366	0.97515	0.00284	0.53940	0.98536	19.07206	0.74616
iTransformer	0.72755†	0.00380†	1.01866†	0.00297†	0.50193†	1.02537†	19.84152†	0.78613†
S-GCN	0.67375†	0.00416†	1.12206†	0.00327†	0.39531†	1.12381†	21.82139†	0.86146†
Synthesizer(Random)	0.73799	0.00373†	1.00028	0.00291	0.50193†	1.02537†	19.84152†	0.78613†
GCN	0.64802†	0.00432†	1.16637†	0.00339†	0.32957†	1.18622†	23.09625†	0.91512†
GAT	0.63661†	0.00439†	1.19921†	0.00349†	0.34812†	1.18150†	23.18125†	0.91086†
GATV2	0.61409†	0.00452†	1.21464†	0.00354†	0.24389†	1.26129†	24.93879†	0.97099†
GKN	**0.76126**	**0.00356**	**0.95538**	**0.00278**	**0.55464**	**0.96210**	**18.69096**	**0.73405**

† marks the variants that GKN outperforms significantly at p-value < 0.05 over paired samples t-test. **Bolded** results indicate the best in each metric. Wavy results indicate the second best in each metric.

(I) Both Input Similarity-based and Learning-based Graph models significantly outperform Fixed Graph models, with improvements ranging from 3.97% to 121.16% higher R^2, 3.75% to 21.88% lower RMSE, 3.80% to 23.52% lower MAPE, and 3.63% to 23.15% lower MAE.

(II) The performance of Input Similarity-based Graph models and Learning-based Graph models is closely matched.

(III) Models utilizing local graphs, such as iTransformer, Flowformer, and Synthesizer(Random), demonstrate superior performance over those employing global graphs like S-GCN, GCN, GAT, and GAT-V2. Specifically, the R^2 values show an improvement ranging from 7.98% to 21.62%, while the RMSE values are reduced by 8.68% to 18.87%. Additionally, the MAPE values exhibit a decrease of 9.22% to 19.72%, and the MAE values are lower by 9.30% to 19.73%.

(IV) For the CA dataset, our GKN model surpasses the baseline models across several metrics, improving R^2 by 1.87% to 24.24%, reducing RMSE by 2.69% to 21.21%, lowering MAPE by 2.10% to 21.64%, and decreasing MAE by 1.99% to 21.54%.

(V) Similarly, on the PR dataset, the GKN model outclasses the competition with a 2.83% to 127.41% enhancement in R^2, a 2.36% to 23.72% reduction in RMSE, a 2.00% to 25.05% decrease in MAPE, and a 1.62% to 24.40% drop in MAE.

Observation (I) suggests that predefining the graph structure, thereby fixing prior knowledge, may constrain model performance, reflecting the necessity of data-driven adjustments to prior assumptions during model training. This supports the assertion that direct incorporation of prior knowledge without adjustment may not be conducive to optimal performance. Observation (II) indicates that the choice between model-driven or similarity-based graph construction does not fundamentally affect performance outcomes. Observation (III) highlights the expressive power of local graph models as a key factor in their superior performance. Finally, Observations (IV) and (V) demonstrate that despite being a local graph-based model, our GKN model significantly outperforms most baselines, underscoring the potential of correctly integrating prior knowledge in the variational Bayesian inference framework to enhance model performance.

5.1.5. *Computational Time*

Building upon the assessment of model performance, we proceeded to analyze the model's training and testing durations. Fig. 7 displays a comparative analysis of computation times across various models. Observations from Fig. 7 reveal that both the training and testing phases of our model necessitate less time than those of the baseline models. This observation underscores the practicality of our model for industrial process modeling, particularly highlighting its advantage in terms of timeliness.

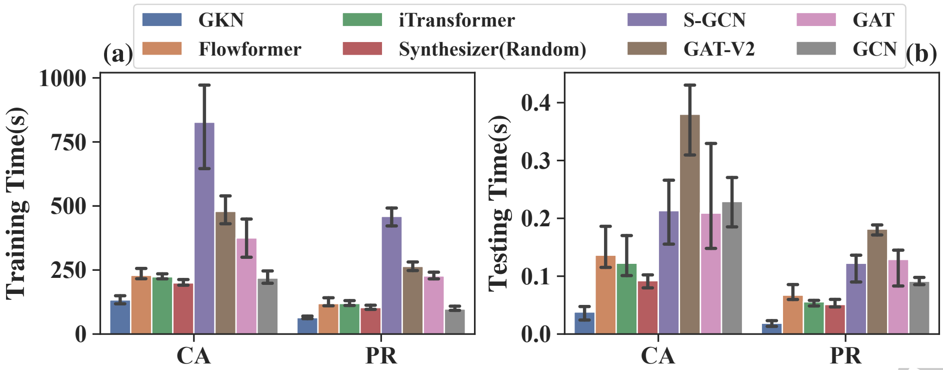

Figure 7 | The computation time comparison (a) training time, (b) testing time.

Table 2 | Ablation Study Results on CA Dataset

Module			R2		RMSE		MAPE		MAE	
AG	PI	MP	Value	Reduction(↓)	Value	Reduction(↑)	Value	Reduction(↑)	Value	Reduction(↑)
✓	✗	✓	0.71515	6.058%	0.00387	8.669%	1.03142	7.959%	0.00300	7.935%
✗	✗	✗	0.73289	3.727%	0.00375	5.421%	0.99585	4.236%	0.00290	4.150%
✗	✓	✓	0.62593	17.778%	0.00424	19.241%	1.11061	16.248%	0.00324	16.471%
✓	✓	✓	0.76126	-	0.00356	-	0.95538	-	0.00278	-

5.2. Sensitivity Analysis

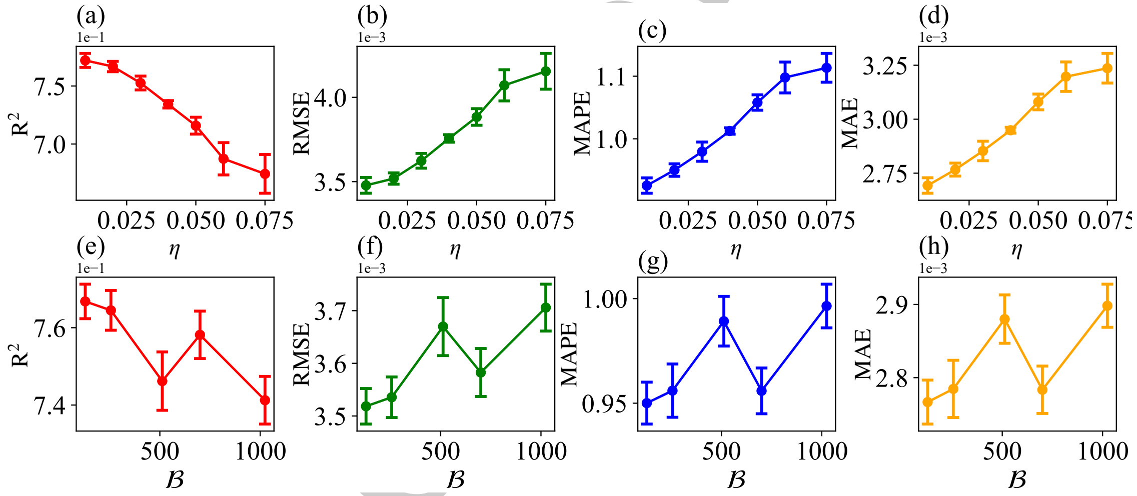

Figure 8 | The monitoring accuracy given varying values of learning rate η (a-d) and batch size $\mathcal{B}$ (e-h) on CA dataset. The panels indicate R^2, RMSE, MAPE and MAE from left to right.

Observations from Figs. 8 and 9 indicate that model performance deteriorates with an increase in the learning rate. This phenomenon can be understood through the lens of simulation error. Referencing (29), learning rate η represents the step size in the SDE simulation process. A larger step size leads to greater simulation error, potentially resulting in optimization outcomes that deviate significantly from the ideal model parameters, thereby impairing model performance.

Conversely, when examining the impact of batch size $\mathcal{B}$, the model's performance exhibits fluctuations across a broad range of batch sizes yet continues to outperform most baseline models. This observation suggests that our model's performance is relatively insensitive to variations in batch size, showcasing

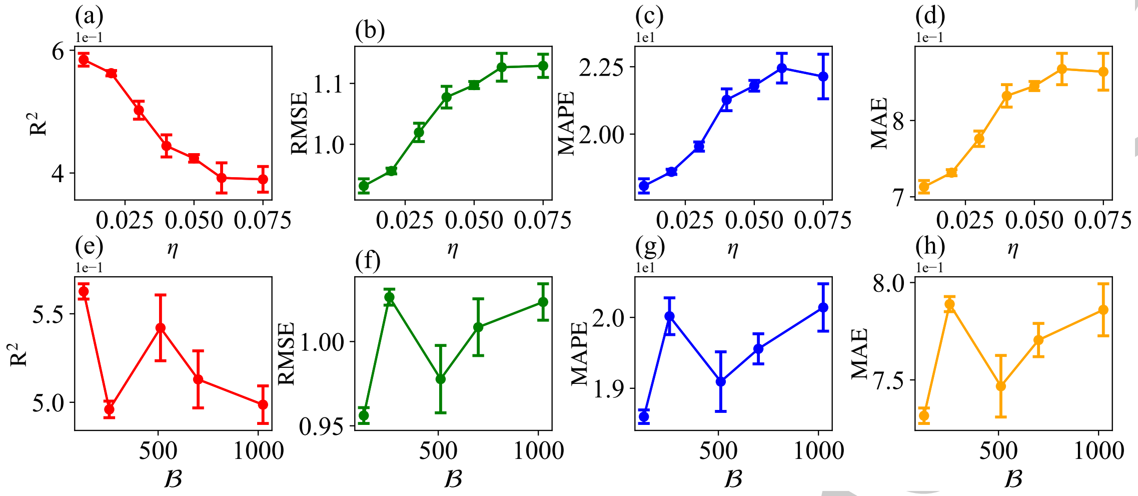

Figure 9 | The monitoring accuracy given varying values of learning rate η (a-d) and batch size $\mathcal{B}$ (e-h) on PR dataset. The panels indicate R^2, RMSE, MAPE and MAE from left to right.

its robustness against such changes. In conclusion, these findings demonstrate that while our model tolerates a wide variety of batch sizes without significant performance degradation. In addition, it is crucial to employ a smaller learning rate during training to ensure optimal performance.

5.3. Ablation Study

Building on the insights gained from the preceding subsection, our objective is to delve deeper into the factors contributing to our model's superior performance. i.e. 'Why does GKN work?'. To this end, we conducted an ablation study focusing on three key components: the learnable knowledge (referred to as 'Adaptive Graph' when $\hat{\mathcal{A}}$ is learnable, abbreviated to 'AG'), the gradient term $\nabla \log p(\hat{\mathcal{A}})$ (denoted as 'Prior Injection', abbreviated to 'PI'), and the message passing block (termed 'Message Passing', abbreviated to 'MP') within the GKN. The findings from this study are presented in Table 2.

Analysis of Table 2 reveals a significant drop in model performance when prior knowledge is fixed, a result that is in alignment with earlier observations noted in Table 1. When prior knowledge is rendered learnable (as shown in Line 1 of the table), there is a marked enhancement in model performance, underscoring the benefit of adaptable graph structures. Despite this improvement, the model does not reach the performance levels of baseline models such as Flowformer, iTransformer, and Synthesizer(Random), indicating that while learnable graphs contribute to improved performance, additional refinements are necessary to surpass these benchmarks. Moreover, ablating the message passing block yields positive results, yet the model still falls short of outperforming the aforementioned baselines. This outcome suggests that while incorporating graph-based models is beneficial, their integration must be executed judiciously to fully leverage their potential. Finally, when we keep the prior knowledge constant, as the third line shows, the model's performance drops sharply. This highlights the problems with the 'correctness' and 'completeness' of prior knowledge. In conclusion, the results of the ablation study affirm the importance of integrating prior knowledge into the training process.

6. Conclusions

In this study, we introduced a novel model termed the GKN, designed to parameterize and harness incomplete knowledge within industrial data-driven modeling for the purpose of achieving automated knowledge integration. Initially, we parameterized the adjacency matrix, which encapsulates the relationships between process variables, using the Bernoulli distribution. This was subsequently transformed into the Dirichlet distribution through an analysis of the message-passing mechanism. We then employed variational inference techniques to approximate the posterior knowledge distribution, encountering challenges with the computation of KL divergence. To overcome these challenges, we reformulated the variational inference problem as a simulation of a SDE, which can be readily implemented in DL backends. To fulfill the normalization requirements imposed by the Dirichlet distribution, we theoretically developed a parameter learning equation that ensures normalization. Additionally, we articulated the model's expressions and detailed the associated algorithms. The efficacy of the GKN model was validated through its application to two inferential sensor tasks.

Nevertheless, there are some issues that remain to be solved. First is the how to promise the iteration process obtain the true posterior sample within the training process and investigate the variance during model training [31, 32]. Extending the proposed approach to other DL structure like CNN, RNN, and Transformer is also a considerable research direction.

Acknowledgement

The first author, Zhichao Chen, expresses his gratitude to Mr. Fangyikang Wang from Zhejiang University for valuable discussions on variational inference, and to Dr. Chang Liu and Dr. Jiaxin Shi of Microsoft for hint on mirror descent.

References

[1] Qingqiang Sun and Zhiqiang Ge. A survey on deep learning for data-driven soft sensors. IEEE Transactions on Industrial Informatics, 17(9):5853–5866, 2021. ISSN 1941-0050. doi: 10.1109/TII.2021.3053128.

[2] Moritz von Stosch, Rui Oliveira, Joana Peres, and Sebastião Feyo de Azevedo. Hybrid semi-parametric modeling in process systems engineering: Past, present and future. Computers & Chemical Engineering, 60:86–101, 2014. ISSN 0098-1354. doi: 10.1016/j.compchemeng.2013.08.008.

[3] Jiaxin Yu, Lingjian Ye, Le Zhou, Zeyu Yang, Feifan Shen, and Zhihuan Song. Dynamic process monitoring based on variational bayesian canonical variate analysis. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 52(4):2412–2422, 2022. doi: 10.1109/TSMC.2021.3051054.

[4] Qingqiang Sun and Zhiqiang Ge. Gated stacked target-related autoencoder: A novel deep feature extraction and layerwise ensemble method for industrial soft sensor application. IEEE Transactions on Cybernetics, 52(5):3457–3468, 2022. doi: 10.1109/TCYB.2020.3010331.

[5] Kangcheng Wang, Chao Shang, Lei Liu, Yongheng Jiang, Dexian Huang, and Fan Yang. Dynamic Soft Sensor Development Based on Convolutional Neural Networks. Industrial & Engineering Chemistry Research, 58(26):11521–11531, July 2019. ISSN 0888-5885. doi: 10.1021/acs.iecr.9b02513.

[6] Le Yao and Zhiqiang Ge. Dynamic features incorporated locally weighted deep learning model for soft sensor development. IEEE Transactions on Instrumentation and Measurement, 70:1–11, 2021. doi: 10.1109/TIM.2021.3073702.

[7] Chao Zhang, Jaswanth Yella, Yu Huang, Xiaoye Qian, Sergei Petrov, Andrey Rzhetsky, and Sthitie Bom. Soft sensing transformer: Hundreds of sensors are worth a single word. In 2021 IEEE International Conference on Big Data (Big Data), pages 1999–2008, 2021. doi: 10.1109/BigData52589.2021.9671925.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In Proceedings of the 31st International Conference on Neural Information Processing Systems(NIPS'17), page 6000–6010, 2017.

[9] Zhiwen Chen, Jiamin Xu, Tao Peng, and Chunhua Yang. Graph convolutional network-based method for fault diagnosis using a hybrid of measurement and prior knowledge. IEEE Transactions on Cybernetics, pages 1–13, 2021. doi: 10.1109/TCYB.2021.3059002.

[10] Zhiyong Wu, Huan Wang, Chang He, Bingjian Zhang, Tao Xu, and Qinglin Chen. The application of physics-informed machine learning in multiphysics modeling in chemical engineering. Industrial & Engineering Chemistry Research, 62(44):18178–18204, 2023.

[11] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems, volume 32, pages 1–12, 2019.

[12] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, et al. Jax: Autograd and xla. Astrophysics Source Code Library, pages ascl–2111, 2021.

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In International Conference on Learning Representations (ICLR), pages 1–8, 2015.

[14] Yao Ma and Jiliang Tang. Deep Learning on Graphs. Cambridge University Press, 2020.

[15] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In International Conference on Learning Representations (ICLR 2016), pages 1–14, 2016.

[16] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Advances in Neural Information Processing Systems 30(NIPS'17), pages 1025–1035, 2017.

[17] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In International Conference on Learning Representations (ICLR 2018), pages 1–12, 2018.

[18] Durk P Kingma, Tim Salimans, and Max Welling. Variational Dropout and the Local Reparameterization Trick. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 28. Curran Associates, Inc., 2015.

[19] Filippo Santambrogio. {Euclidean, metric, and Wasserstein} gradient flows: an overview. Bulletin of Mathematical Sciences, 7:87–154, 2017.

[20] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. GNNExplainer: Generating explanations for graph neural networks. In Advances in Neural Information Processing Systems 32 (NIPS'19), volume 32, pages 9244–9255, 2019.

[21] Simo Särkkä and Arno Solin. Applied stochastic differential equations, volume 10. Cambridge University Press, 2019. doi: https://doi.org/10.1017/9781108186735.

[22] Fangyikang Wang, Huminhao Zhu, Chao Zhang, Hanbin Zhao, and Hui Qian. Gad-pvi: A general accelerated dynamic-weight particle-based variational inference framework. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 37, pages 1–29, 2023.

[23] Weiming Liu, Xiaolin Zheng, Jiajie Su, Longfei Zheng, Chaochao Chen, and Mengling Hu. Contrastive proxy kernel stein path alignment for cross-domain cold-start recommendation. IEEE Transactions on Knowledge and Data Engineering, 35(11):11216–11230, 2023. doi: 10.1109/TKDE.2022.3233789.

[24] Haixu Wu, Jialong Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Flowformer: Linearizing transformers with conservation flows. In International Conference on Machine Learning, pages 24226–24242. PMLR, 2022.

[25] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. iTransformer: Inverted transformers are effective for time series forecasting. In The Twelfth International Conference on Learning Representations, pages 1–22, 2024. URL https://openreview.net/forum?id=JePfAI8fah.

[26] Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention for transformer models. In International Conference on Machine Learning, pages 10183–10192. PMLR, 2021.

[27] Yalin Wang, Qingkai Sui, Chenliang Liu, Kai Wang, Xiaofeng Yuan, and Guangfeng Dong. Interpretable prediction modeling for froth flotation via stacked graph convolutional network. IEEE Transactions on Artificial Intelligence, 5(1):334–345, 2024. doi: 10.1109/TAI.2023.3240114.

[28] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In International Conference on Learning Representations, 2016.

[29] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In International Conference on Learning Representations, pages 1–12, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

[30] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In International Conference on Learning Representations, pages 1–26, 2022. URL https://openreview.net/forum?id=F72ximsx7C1.

[31] Ming Xu, Matias Quiroz, Robert Kohn, and Scott A Sisson. Variance reduction properties of the reparameterization trick. In The 22nd international conference on artificial intelligence and statistics, pages 2711–2720. PMLR, 2019.

[32] Michael Zhu, Chang Liu, and Jun Zhu. Variance reduction and quasi-newton for particle-based variational inference. In International Conference on Machine Learning, pages 11576–11587. PMLR, 2020.